Docket No. 219061US2SRD

3-11-03

jc978 U.S. PTO 10/059185 01/31/02

## IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

IN RE APPLICATION OF: Noritaka KAWAKATSU

GAU:

SERIAL NO: NEW APPLICATION

EXAMINER:

FILED: HEREWITH

FOR: METHOD AND COMPUTER PROGRAM PRODUCT FOR SOFTWARE/HARDWARE LANGUAGE MODEL CONVERSION

## REQUEST FOR PRIORITY

ASSISTANT COMMISSIONER FOR PATENTS
WASHINGTON, D.C. 20231

SIR:

☐ Full benefit of the filing date of U.S. Application Serial Number , filed , is claimed pursuant to the provisions of **35 U.S.C. §120**.

☐ Full benefit of the filing date of U.S. Provisional Application Serial Number , filed , is claimed pursuant to the provisions of **35 U.S.C. §119(e)**.

☒ Applicants claim any right to priority from any earlier filed applications to which they may be entitled pursuant to the provisions of **35 U.S.C. §119**, as noted below.

In the matter of the above-identified application for patent, notice is hereby given that the applicants claim as priority:

| COUNTRY | APPLICATION NUMBER | MONTH/DAY/YEAR |
|---|---|---|
| Japan | 2001-024887 | January 31, 2001 |

Certified copies of the corresponding Convention Application(s)

☒ are submitted herewith

☐ will be submitted prior to payment of the Final Fee

☐ were filed in prior application Serial No. filed

☐ were submitted to the International Bureau in PCT Application Number
Receipt of the certified copies by the International Bureau in a timely manner under PCT Rule 17.1(a) has been acknowledged as evidenced by the attached PCT/IB/304.

☐ (A) Application Serial No.(s) were filed in prior application Serial No. filed ; and

☐ (B) Application Serial No.(s)

- ☐ are submitted herewith
- ☐ will be submitted prior to payment of the Final Fee

Respectfully Submitted,

OBLON, SPIVAK, McCLELLAND,
MAIER & NEUSTADT, P.C.

Marvin J. Spivak
Registration No. 24,913

James D. Hamilton
Registration No. 28,421

22850
Tel. (703) 413-3000
Fax. (703) 413-2220
(OSMMN 10/98)

01S1543



# 日　本　国　特　許　庁
# JAPAN PATENT OFFICE

別紙添付の書類に記載されている事項は下記の出願書類に記載されている事項と同一であることを証明する。

This is to certify that the annexed is a true copy of the following application as filed with this Office

出願年月日
Date of Application: ２００１年　１月３１日

出願番号
Application Number: 特願２００１－０２４８８７

出願人
Applicant(s): 株式会社東芝

２００１年１２月１４日

特許庁長官
Commissioner,
Japan Patent Office 及川耕造

出証番号　出証特２００１－３１０８９００

【書類名】　特許願
【整理番号】　A000100062
【提出日】　平成13年 1月31日
【あて先】　特許庁長官　殿
【国際特許分類】　G06F 15/00
【発明の名称】　ソフトウェア・ハードウェア言語モデル変換装置及び方法並びにシステム設計支援装置
【請求項の数】　17
【発明者】
【住所又は居所】　神奈川県川崎市幸区小向東芝町１番地　株式会社東芝研究開発センター内
【氏名】　川勝　則孝
【特許出願人】
【識別番号】　000003078
【氏名又は名称】　株式会社　東芝
【代理人】
【識別番号】　100058479
【弁理士】
【氏名又は名称】　鈴江　武彦
【電話番号】　03-3502-3181
【選任した代理人】
【識別番号】　100084618
【弁理士】
【氏名又は名称】　村松　貞男
【選任した代理人】
【識別番号】　100068814
【弁理士】
【氏名又は名称】　坪井　淳

【選任した代理人】

【識別番号】　100092196

【弁理士】

【氏名又は名称】　橋本　良郎

【選任した代理人】

【識別番号】　100091351

【弁理士】

【氏名又は名称】　河野　哲

【選任した代理人】

【識別番号】　100088683

【弁理士】

【氏名又は名称】　中村　誠

【選任した代理人】

【識別番号】　100070437

【弁理士】

【氏名又は名称】　河井　将次

【手数料の表示】

【予納台帳番号】　011567

【納付金額】　21,000円

【提出物件の目録】

【物件名】　明細書　1

【物件名】　図面　1

【物件名】　要約書　1

【プルーフの要否】　要

【書類名】　明細書

【発明の名称】　ソフトウェア・ハードウェア言語モデル変換装置及び方法並びにシステム設計支援装置

【特許請求の範囲】

【請求項１】

ソフトウェア記述言語により記述された第１のモデルをハードウェア記述言語により記述された第２のモデルに変換するソフトウェア・ハードウェア言語モデル変換装置であって、

同一の共有変数へ書き込みを行う複数の並列手続きが前記第１のモデルに含まれているか否かを考慮せずに、前記第１のモデルを前記第２のモデルに変換する変換手段と、

この変換手段により得られた前記第２のモデルに、前記同一の共有変数へ書き込みを行う複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出する検出手段と、

前記検出手段により前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、データ信号及び代入タイミング信号の組を入力し、該データ信号のうち該代入タイミング信号が変化したプロセスに対応するものを、前記共有変数の値を保持する信号へ出力する値解決プロセスを生成する値解決プロセス生成手段とを備えたことを特徴とするソフトウェア・ハードウェア言語モデル変換装置。

【請求項２】

前記値解決プロセス生成手段は、前記変換手段により得られた前記第２のモデルがコンポーネント階層を持つものである場合、前記値解決プロセスを該コンポーネント階層に応じて階層毎に生成することを特徴とする請求項１に記載のソフトウェア・ハードウェア言語モデル変換装置。

【請求項３】

前記値解決プロセス生成手段は、前記同一の共有変数に対する値解決プロセスが複数階層に存在する場合に、下位に位置する値解決プロセスについては、該値解決プロセスから、前記データ信号に加えて、該値解決プロセスから出力する該

データ信号の値が更新されるときに変化する代入タイミング信号をも出力させ、該データ信号及び該代入タイミング信号を上位の値解決プロセス接続することを特徴とする請求項２に記載のソフトウェア・ハードウェア言語モデル変換装置。

【請求項４】

前記値解決プロセス生成手段は、前記同一の共有変数に対する値解決プロセスが複数階層に存在する場合に、最も上位に位置する値解決プロセスのみを残して他の値解決プロセスを削除し、削除した値解決プロセスに接続されていた前記並列プロセスからの前記データ信号及び該代入タイミング信号は、該最も上位に位置する値解決プロセスに接続することを特徴とする請求項２または３記載のソフトウェア・ハードウェア言語モデル変換装置。

【請求項５】

前記値解決プロセス生成手段により生成された前記値解決プロセスのうち、それが対象とするプロセスが、同一の手続きを呼び出す複数の並列手続きに相当する複数の並列プロセスであるものについては、該値解決プロセスを、各プロセスから、前記同一の手続きに相当するプロセスを呼び出す実行起動トリガー信号及び引数信号の組を入力とし、該呼び出されたプロセスからの終了信号及び戻り値信号を各プロセスへの出力とし、同時に複数のプロセスからの実行起動トリガー信号が立ち上がった場合に該プロセスの呼び出しを排他制御する手続き呼び出し解決プロセスに変換するプロセス変換手段を更に備えたことを特徴とする請求項１ないし４のいずれか１項に記載のソフトウェア・ハードウェア言語モデル変換装置。

【請求項６】

ソフトウェア記述言語により記述された第１のモデルをハードウェア記述言語により記述された第２のモデルに変換するソフトウェア・ハードウェア言語モデル変換装置であって、

同一の手続きを呼び出す複数の並列手続きが前記第１のモデルに含まれているか否かを考慮せずに、前記第１のモデルを前記第２のモデルに変換する変換手段と、

この変換手段により得られた前記第２のモデルに、前記同一の手続きを呼び出

す複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出する検出手段と、

前記検出手段により前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、前記同一の手続きに相当するプロセスを呼び出す実行起動トリガー信号及び引数信号の組を入力とし、該呼び出されたプロセスからの終了信号及び戻り値信号を各プロセスへの出力とし、同時に複数のプロセスからの実行起動トリガー信号が立ち上がった場合に該プロセスの呼び出しを排他制御する手続き呼び出し解決プロセスを生成する手続き呼び出し解決プロセス生成手段とを備えたことを特徴とするソフトウェア・ハードウェア言語モデル変換装置。

【請求項７】

前記手続き呼び出し解決プロセス生成手段は、前記変換手段により得られた前記第２のモデルがコンポーネント階層を持つものである場合、前記手続き呼び出し解決プロセスを該コンポーネント階層に応じて階層毎に生成することを特徴とする請求項６に記載のソフトウェア・ハードウェア言語モデル変換装置。

【請求項８】

前記手続き呼び出し解決プロセス生成手段は、同一のプロセスに対する手続き呼び出し解決プロセスが複数階層に存在する場合に、下位に位置する手続き呼び出し解決プロセスを、上位の手続き呼び出し解決プロセスに接続することを特徴とする請求項７に記載のソフトウェア・ハードウェア言語モデル変換装置。

【請求項９】

前記手続き呼び出し解決プロセス生成手段は、同一のプロセスに対する手続き呼び出し解決プロセスが複数階層に存在する場合に、最も上位に位置する手続き呼び出し解決プロセスのみを残して他の手続き呼び出し解決プロセスを削除し、削除した手続き呼び出し解決プロセスに接続されていた前記並列プロセスを、該最も上位に位置する手続き呼び出し解決プロセスに接続することを特徴とする請求項７または８記載のソフトウェア・ハードウェア言語モデル変換装置。

【請求項１０】

ソフトウェア記述言語により記述された第１のモデルをハードウェア記述言語

により記述された第２のモデルに変換するソフトウェア・ハードウェア言語モデル変換方法であって、

同一の共有変数へ書き込みを行う複数の並列手続きが前記第１のモデルに含まれているか否かを考慮せずに、前記第１のモデルを前記第２のモデルに変換し、

これにより得られた前記第２のモデルに、前記同一の共有変数へ書き込みを行う複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出し、

これにより前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、データ信号及び代入タイミング信号の組を入力し、該データ信号のうち該代入タイミング信号が変化したプロセスに対応するものを、前記共有変数の値を保持する信号へ出力する値解決プロセスを生成することを特徴とするソフトウェア・ハードウェア言語モデル変換方法。

【請求項１１】

ソフトウェア記述言語により記述された第１のモデルをハードウェア記述言語により記述された第２のモデルに変換するソフトウェア・ハードウェア言語モデル変換方法であって、

同一の手続きを呼び出す複数の並列手続きが前記第１のモデルに含まれているか否かを考慮せずに、前記第１のモデルを前記第２のモデルに変換し、

これにより得られた前記第２のモデルに、前記同一の手続きを呼び出す複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出し、

これにより前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、前記同一の手続きに相当するプロセスを呼び出す実行起動トリガー信号及び引数信号の組を入力とし、該呼び出されたプロセスからの終了信号及び戻り値信号を各プロセスへの出力とし、同時に複数のプロセスからの実行起動トリガー信号が立ち上がった場合に該プロセスの呼び出しを排他制御する手続き呼び出し解決プロセスを生成することを特徴とするソフトウェア・ハードウェア言語モデル変換方法。

【請求項１２】

ソフトウェア記述言語により記述された第１のモデルをハードウェア記述言語により記述された第２のモデルに変換するソフトウェア・ハードウェア言語モデ

ル変換装置としてコンピュータを機能させるためのプログラムを記録したコンピュータ読取り可能な記録媒体であって、

同一の共有変数へ書き込みを行う複数の並列手続きが前記第１のモデルに含まれているか否かを考慮せずに、前記第１のモデルを前記第２のモデルに変換する変換機能と、

この変換機能により得られた前記第２のモデルに、前記同一の共有変数へ書き込みを行う複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出する検出機能と、

前記検出機能により前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、データ信号及び代入タイミング信号の組を入力し、該データ信号のうち該代入タイミング信号が変化したプロセスに対応するものを、前記共有変数の値を保持する信号へ出力する値解決プロセスを生成する値解決プロセス生成機能とをコンピュータに実現させるためのプログラムを記録したコンピュータ読取り可能な記録媒体。

【請求項１３】

ソフトウェア記述言語により記述された第１のモデルをハードウェア記述言語により記述された第２のモデルに変換するソフトウェア・ハードウェア言語モデル変換装置としてコンピュータを機能させるためのプログラムを記録したコンピュータ読取り可能な記録媒体であって、

同一の手続きを呼び出す複数の並列手続きが前記第１のモデルに含まれているか否かを考慮せずに、前記第１のモデルを前記第２のモデルに変換する変換機能と、

この変換機能により得られた前記第２のモデルに、前記同一の手続きを呼び出す複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出する検出機能と、

前記検出機能により前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、前記同一の手続きに相当するプロセスを呼び出す実行起動トリガー信号及び引数信号の組を入力とし、該呼び出されたプロセスからの終了信号及び戻り値信号を各プロセスへの出力とし、同

時に複数のプロセスからの実行起動トリガー信号が立ち上がった場合に該プロセスの呼び出しを排他制御する手続き呼び出し解決プロセスを生成する手続き呼び出し解決プロセス生成機能とをコンピュータに実現させるためのプログラムを記録したコンピュータ読取り可能な記録媒体。

【請求項１４】

ソフトウェア記述言語により記述された第１のモデルをハードウェア記述言語により記述された第２のモデルに変換するソフトウェア・ハードウェア言語モデル変換装置としてコンピュータを機能させるためのプログラムであって、

同一の共有変数へ書き込みを行う複数の並列手続きが前記第１のモデルに含まれているか否かを考慮せずに、前記第１のモデルを前記第２のモデルに変換する変換機能と、

この変換機能により得られた前記第２のモデルに、前記同一の共有変数へ書き込みを行う複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出する検出機能と、

前記検出機能により前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、データ信号及び代入タイミング信号の組を入力し、該データ信号のうち該代入タイミング信号が変化したプロセスに対応するものを、前記共有変数の値を保持する信号へ出力する値解決プロセスを生成する値解決プロセス生成機能とをコンピュータに実現させるためのプログラム。

【請求項１５】

ソフトウェア記述言語により記述された第１のモデルをハードウェア記述言語により記述された第２のモデルに変換するソフトウェア・ハードウェア言語モデル変換装置としてコンピュータを機能させるためのプログラムであって、

同一の手続きを呼び出す複数の並列手続きが前記第１のモデルに含まれているか否かを考慮せずに、前記第１のモデルを前記第２のモデルに変換する変換機能と、

この変換機能により得られた前記第２のモデルに、前記同一の手続きを呼び出す複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出する検

出機能と、

前記検出機能により前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、前記同一の手続きに相当するプロセスを呼び出す実行起動トリガー信号及び引数信号の組を入力とし、該呼び出されたプロセスからの終了信号及び戻り値信号を各プロセスへの出力とし、同時に複数のプロセスからの実行起動トリガー信号が立ち上がった場合に該プロセスの呼び出しを排他制御する手続き呼び出し解決プロセスを生成する手続き呼び出し解決プロセス生成機能とをコンピュータに実現させるためのプログラム。

【請求項１６】

計算に関する仕様および通信に関する仕様を含むシステム仕様モデルを設計するための仕様モデル記述手段と、

システム仕様モデルにおいて、特定のアーキテクチャに対して、システム仕様モデルの内容を保ったまま構造を分配して計算に関する仕様および通信に関する仕様の一部分をアーキテクチャに割り当てるためのアーキテクチャ探索手段と、

システム仕様モデルにおいて、割り当てられた特定のアーキテクチャ上の仕様要素間の通信手順を合成するためのコミュニケーション合成手段と、

システム仕様モデルからハードウェア仕様モデルを生成するためのハードウェア仕様生成手段と、

システム仕様モデルからソフトウェア仕様モデルを生成するためのソフトウェア仕様生成手段とを備え、

前記ハードウェア仕様生成手段は、請求項１ないし９のいずれか１項に記載のソフトウェア・ハードウェア言語モデル変換装置に相当する機能を含むものであることを特徴とするシステム設計支援装置。

【請求項１７】

前記仕様モデル記述手段、前記アーキテクチャ探索手段または前記コミュニケーション合成手段の少なくとも一つにおいて、システム仕様モデルを部品化し、これを設計に再利用するための部品化・再利用手段を更に備えたことを特徴とする請求項１６に記載のシステム設計支援装置。

【発明の詳細な説明】

【０００１】

【発明の属する技術分野】

本発明は、ソフトウェア記述をハードウェア動作記述に自動変換するソフトウェア・ハードウェア言語モデル変換装置及び方法並びにシステム設計支援装置に関する。

【０００２】

【従来の技術】

昨今、ＣやＣ++さらには仕様記述言語ＳｐｅｃＣ（“SpecC:Specification Language and Methodology”,Daniel D.Gajski,Kluwer Academic Publishers,Dordrecht,ISBN0-7923-7822-9に詳しい)といったソフトウェア記述言語で、システム全体あるいはハードウェアを設計することが注目されている。すなわち、ソフトウェア記述言語により仕様を記述し、これを、ハードウェアの詳細設計に移行する際に下流の論理合成ツールなどに接続するため、ＶＨＤＬやＶｅｒｉｌｏｇ等のハードウェア記述言語による仕様記述に変換するものである。

【０００３】

Ｃ言語からハードウェア言語モデルを生成するツールは、すでに製品化されているものもある。しかし、そのときに入力するコードはシーケンシャルな処理を記述したものに限られており、部分的な設計においては役に立つものの、より大規模な設計を行おうとすると、シーケンシャルな仕様では書ききれない部分があるので不都合がある。従って、並列プログラムをサポートできるソフトウェア・ハードウェア言語モデル変換が望まれている。

【０００４】

【発明が解決しようとする課題】

ソフトウェア記述言語による記述で並列プログラムを含むとき、その記述を動作レベルのハードウェア記述言語に変換する場合、共有変数の取り扱いを素直に変換することができないという問題点があった。つまり、ソフトウェアでは代入が実行される順序に従って値が決まるが、ハードウェア記述においては各並列プログラムからの信号出力を単純に結線すると値が不定または代入順序とは関係なく決まる値となってしまう。そこで、システマティックに共有変数への代入処理

をハードウェア記述へ変換を行う方法が望まれる。

【０００５】

また、ソフトウェア記述言語による記述で並列に動作し得る手続きの呼び出しを含むとき、その記述を動作レベルのハードウェア記述言語に変換する場合、同一の手続きに対する呼び出しの取り扱いを素直に変換することができないという問題点があった。つまり、並列手続きをプロセスへ置きかえた場合、同時に同じ手続きが呼び出され得るが、素直にプロセス間の信号によるシェークハンドで表すと、排他的呼び出しにならないことになってしまう。そこで、システマティックに排他的呼び出しの機構をコード生成する方法が望まれる。

【０００６】

本発明は、上記事情を考慮してなされたもので、並列プログラムとその間に共有する変数を含むソフトウェア記述を同様の動作をする動作レベルのハードウェア記述に変換することのできるソフトウェア・ハードウェア言語モデル変換装置及び方法並びにシステム設計支援装置を提供することを目的とする。

【０００７】

また、本発明は、ソフトウェア言語による並列プログラムをハードウェア記述に変換する際、手続き呼び出しに相当する排他的なプロセス実行処理を実現することのできるソフトウェア・ハードウェア言語モデル変換装置及び方法並びにシステム設計支援装置を提供することを目的とする。

【０００８】

【課題を解決するための手段】

本発明は、ソフトウェア記述言語により記述された第１のモデルをハードウェア記述言語により記述された第２のモデルに変換するソフトウェア・ハードウェア言語モデル変換装置であって、同一の共有変数へ書き込みを行う複数の並列手続きが前記第１のモデルに含まれているか否かを考慮せずに、前記第１のモデルを前記第２のモデルに変換する変換手段と、この変換手段により得られた前記第２のモデルに、前記同一の共有変数へ書き込みを行う複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出する検出手段と、前記検出手段により前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全

部又は一部の各プロセスから、データ信号及び代入タイミング信号の組を入力し、該データ信号のうち該代入タイミング信号が変化したプロセスに対応するものを、前記共有変数の値を保持する信号へ出力する値解決プロセスを生成する値解決プロセス生成手段とを備えたことを特徴とする。

【０００９】

これによって、並列プログラムとその間に共有する変数を含むソフトウェア記述を同様の動作をする動作レベルのハードウェア記述に変換するができるようになる。

【００１０】

好ましくは、前記値解決プロセス生成手段により生成された前記値解決プロセスのうち、それが対象とするプロセスが、同一の手続きを呼び出す複数の並列手続きに相当する複数の並列プロセスであるものについては、該値解決プロセスを、各プロセスから、前記同一の手続きに相当するプロセスを呼び出す実行起動トリガー信号及び引数信号の組を入力とし、該呼び出されたプロセスからの終了信号及び戻り値信号を各プロセスへの出力とし、同時に複数のプロセスからの実行起動トリガー信号が立ち上がった場合に該プロセスの呼び出しを排他制御する（例えば、予め定められた順番で実行する）手続き呼び出し解決プロセスに変換するプロセス変換手段を更に備えるようにしてもよい。

【００１１】

これによって、ソフトウェア言語による並列プログラムをハードウェア記述に変換する際、手続き呼び出しに相当する排他的なプロセス実行処理を実現することができるようになる。

【００１２】

また、本発明は、ソフトウェア記述言語により記述された第１のモデルをハードウェア記述言語により記述された第２のモデルに変換するソフトウェア・ハードウェア言語モデル変換装置であって、同一の手続きを呼び出す複数の並列手続きが前記第１のモデルに含まれているか否かを考慮せずに、前記第１のモデルを前記第２のモデルに変換する変換手段と、この変換手段により得られた前記第２のモデルに、前記同一の手続きを呼び出す複数の並列手続きに相当する複数の並

列プロセスが存在するか否か検出する検出手段と、前記検出手段により前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、前記同一の手続きに相当するプロセスを呼び出す実行起動トリガー信号及び引数信号の組を入力とし、該呼び出されたプロセスからの終了信号及び戻り値信号を各プロセスへの出力とし、同時に複数のプロセスからの実行起動トリガー信号が立ち上がった場合に該プロセスの呼び出しを排他制御する（例えば、予め定められた順番で実行する）手続き呼び出し解決プロセスを生成する手続き呼び出し解決プロセス生成手段とを備えたことを特徴とする。

【００１３】

これによって、ソフトウェア言語による並列プログラムをハードウェア記述に変換する際、手続き呼び出しに相当する排他的なプロセス実行処理を実現することができるようになる。

【００１４】

また、本発明は、ソフトウェア記述言語により記述された第１のモデルをハードウェア記述言語により記述された第２のモデルに変換するソフトウェア・ハードウェア言語モデル変換方法であって、同一の共有変数へ書き込みを行う複数の並列手続きが前記第１のモデルに含まれているか否かを考慮せずに、前記第１のモデルを前記第２のモデルに変換し、これにより得られた前記第２のモデルに、前記同一の共有変数へ書き込みを行う複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出し、これにより前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、データ信号及び代入タイミング信号の組を入力し、該データ信号のうち該代入タイミング信号が変化したプロセスに対応するものを、前記共有変数の値を保持する信号へ出力する値解決プロセスを生成することを特徴とする。

【００１５】

また、本発明は、ソフトウェア記述言語により記述された第１のモデルをハードウェア記述言語により記述された第２のモデルに変換するソフトウェア・ハードウェア言語モデル変換方法であって、同一の手続きを呼び出す複数の並列手続きが前記第１のモデルに含まれているか否かを考慮せずに、前記第１のモデルを

前記第２のモデルに変換し、これにより得られた前記第２のモデルに、前記同一の手続きを呼び出す複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出し、これにより前記複数の並列プロセスが検出された場合に、検出された該並列プロセスの全部又は一部の各プロセスから、前記同一の手続きに相当するプロセスを呼び出す実行起動トリガー信号及び引数信号の組を入力とし、該呼び出されたプロセスからの終了信号及び戻り値信号を各プロセスへの出力とし、同時に複数のプロセスからの実行起動トリガー信号が立ち上がった場合に該プロセスの呼び出しを排他制御する手続き呼び出し解決プロセスを生成することを特徴とする。

【００１６】

また、本発明に係るシステム設計支援装置は、計算に関する仕様および通信に関する仕様を含むシステム仕様モデルを設計するための仕様モデル記述手段と、特定のアーキテクチャに対しシステム仕様モデルの内容を保ったまま当該システム仕様モデルの持つ構造を分配して、当該システム仕様モデルに含まれる計算に関する仕様および通信に関する仕様の所定部分を当該特定のアーキテクチャに割り当てるためのアーキテクチャ探索手段と、特定のアーキテクチャに対し仕様要素が割り当てられたシステム仕様モデルにつき、当該割り当てられた特定のアーキテクチャ上の仕様要素間の通信手順を合成するためのコミュニケーション合成手段と、システム仕様モデルをもとにハードウェア仕様モデルを生成するためのハードウェア仕様生成手段と、システム仕様モデルをもとにソフトウェア仕様モデルを生成するためのソフトウェア仕様生成手段とを備え、前記ハードウェア仕様生成手段は、請求項１ないし９のいずれか１項に記載のソフトウェア・ハードウェア言語モデル変換装置に相当する機能を含むものであることを特徴とする。

【００１７】

なお、装置に係る本発明は方法に係る発明としても成立し、方法に係る本発明は装置に係る発明としても成立する。

また、装置または方法に係る本発明は、コンピュータに当該発明に相当する手順を実行させるための（あるいはコンピュータを当該発明に相当する手段として機能させるための、あるいはコンピュータに当該発明に相当する機能を実現させ

るための）プログラムとしても成立し、該プログラムを記録したコンピュータ読取り可能な記録媒体としても成立する。

【００１８】

【発明の実施の形態】

以下、図面を参照しながら発明の実施の形態を説明する。

【００１９】

図１８に、本発明の実施の形態に係るシステム設計支援装置１００の構成例を示す。図１８に示されるように、システム設計支援装置１００は、仕様モデル記述部１０１、アーキテクチャ探索部１０２、コミュニケーション合成部１０３、ハードウェア仕様生成部１０４、部品化・再利用部１０５、ソフトウェア仕様生成部１０６、システム仕様記録部１０７を備えている。システム設計支援装置１００は、例えば、コンピュータを用いて構成可能であり、この場合、各処理部分はプログラムによって実施できる。

【００２０】

システム仕様記録部１０７は、入力データや各処理部の処理結果の全部又は一部、例えば、計算機で実行されるソフトウェアの仕様や、半導体を組み合わせたハードウェアの仕様や、ソフトウェアとハードウェアを組み合わせた組み込みシステムの仕様や、ワークフロー等のビジネスプロセスの仕様に対しシステムレベルでの仕様等や、その他の必要なデータを記録するためのものである。

【００２１】

仕様モデル記述部１０１は、設計者が予め定められた仕様記述形式に従って計算内容の仕様、通信内容の仕様を記述することを支援するためのものである。仕様モデル記述部１０１の支援の結果、仕様モデルが生成される。仕様記述形式には、例えば、構造化プログラミング言語に代表される構造化テキスト形式や、グラフを利用する構造図形式や、テーブルを利用する表形式が有る。

【００２２】

アーキテクチャ探索部１０２は、与えられた仕様記述モデルと、アーキテクチャ（例えば、ハードウェア実行環境とソフトウェア実行環境の構成）に対し、仕様の内容を保存したまま仕様モデルの部分構造を分割しアーキテクチャ要素に分

配する。すなわち、仕様モデル記述部１０１で設計された計算内容の仕様や通信内容の仕様を構成する部品をアーキテクチャ要素に割り当てる。

【００２３】

コミュニケーション合成部１０３は、アーキテクチャ探索部１０２で分配された通信仕様要素間に対して通信手順（プロトコル）を挿入し、通信内容の仕様が挿入された通信手順と整合性を保つためのプロトコル変換（通信手順の組替え）を行う。

【００２４】

なお、アーキテクチャ探索部１０２が処理対象とする仕様記述モデルは、仕様モデル記述部１０１を利用して記述されたものに限定されず、他のツール等で記述されたものでもよい。また、コミュニケーション合成部１０３が処理対象とする仕様記述モデルは、アーキテクチャ探索部１０２の出力結果（またはこの結果に修正を加えたもの）に限定されず、他のツール等で記述されたものでもよい。

【００２５】

ハードウェア仕様生成部１０４は、システム仕様からハードウェア記述言語などによるハードウェア仕様を生成する。

【００２６】

ソフトウェア仕様生成部１０６は、システム仕様からソフトウェア記述言語などによるソフトウェア仕様を生成する。

【００２７】

ハードウェア仕様生成部１０４やソフトウェア仕様生成部１０６は、仕様モデル記述部１０１の出力結果（またはこの結果に修正を加えたもの）、アーキテクチャ探索部１０２の出力結果（またはこの結果に修正を加えたもの）、コミュニケーション合成部１０３の出力結果（またはこの結果に修正を加えたもの）、あるいは他のツール等で記述されたものなどを処理対象とすることができる。

【００２８】

部品化・再利用部１０５は、仕様モデル記述部１０１、アーキテクチャ探索部１０２、コミュニケーション合成部１０３の全部または一部の設計段階において、仕様を部品化し、これを設計に再利用するためのものである。

【００２９】

以下詳述するソフトウェア・ハードウェア言語モデル変換装置は、図１８のシステム設計支援装置のハードウェア仕様生成部１０４に用いられるもので、システム仕様からＨＤＬ記述を生成することを可能にするものである。

【００３０】

図１に、本発明の一実施形態に係るソフトウェア・ハードウェア言語モデル変換装置（以下、言語モデル変換装置と呼ぶ）の構成例を示す。

【００３１】

図１に示されるように、本言語モデル変換装置は、骨格コード生成部１、コード置換部２、値解決プロセス生成部３を備えている。

【００３２】

本言語モデル変換装置は、ＳｐｅｃＣ言語等のソフトウェア記述言語により記述されたモデルから、ＶＨＤＬ等のハードウェア記述言語により記述された（同様の動作をする）モデルを生成するものである。

【００３３】

まず、図２（ソフトウェア記述言語モデルの説明図）および図３（ハードウェア記述言語の説明図）を参照して、本言語モデル変換装置の概要について説明する。なお、図２および図３において、１０２１は共有変数、１０２３は並列に動作し得る手続き、１０２２は手続きからの出力、１０１１は共有変数の値を保持する信号、１０１５は互いに並行に実行されるプロセス、１０１２は本言語モデル変換装置が生成する値解決プロセス（矢印が信号の入出力を表す）、１０１３はプロセスの代入データ信号、１０１４はプロセスの代入タイミング信号である。

【００３４】

図２のようにソフトウェア記述言語において並列プログラム間の共有変数がある場合、本言語モデル変換装置は、図３のようにハードウェア記述言語において、各共有変数に対して、値保持信号（１０１１）とその共有変数に書き込みアクセスし得る各並列プログラム（１０２３）に対応するプロセス（１０１５）ごとに、代入データ信号（１０１３）と代入タイミング信号（１０１４）を生成し、

代入タイミング信号（１０１４）の変化に従って、値保持信号（１０１１）への出力を、代入タイミング信号（１０１４）と組になるデータ信号（１０１３）に切り替える、値解決プロセス（１０１２）を生成する。値解決プロセス（１０１２）の生成は、入力コードにおけるその共有変数代入の処理に対応する部分にデータ信号（１０１３）への代入と代入タイミング信号（１０１４）の変化を与える処理（図８参照）を生成することによって行う。

【００３５】

また、ここでは、図４のように入力されるソフトウェア記述言語モデルにおいて手続きが階層化されていて出力されるハードウェア記述言語モデルにおいて並列プロセスが階層化されている場合、図５のように値解決プロセスは階層ごとにあって信号がその階層に従って接続される。その場合、下位階層の値解決プロセスの出力には、代入タイミング信号が加わる。階層があるときは、値反映の待ちを挿入する。

【００３６】

以下、本言語モデル変換装置の処理について説明する。

【００３７】

図６に、本言語モデル変換装置の処理手順の一例を示す。図６に示されるように本手順は、骨格コード生成処理（処理１～６）と、コード置換生成処理（処理７，８）と、値解決プロセス生成処理（処理９，１０）とを行うものである。

【００３８】

なお、後述する処理１～１０は、図７に示すような実行順序制約（矢印の終点側にある処理はその矢印の起点側にある全処理の終了後に実行する）を満たす限り、どのような順番で処理を行っても構わない。また、複数の処理を並列に行っても良い。

【００３９】

ここでは、ＳｐｅｃＣ言語のソースコードを入力して、ＶＨＤＬソースコードを生成する場合を例にとって説明する。また、出力するＶＨＤＬコードは、ＣＬＫという信号を各Ｅｎｔｉｔｙで ＩＮ属性の入力ポート信号として持ち、その入力値は同期クロックであると仮定しているものとして説明する。

【００４０】

＜骨格コード生成処理（骨格コード生成部１）＞

（処理１）ＳｐｅｃＣ言語の仕様は、Ｃ言語のグローバル関数・グローバル変数とｃｌａｓｓと呼ばれる要素からなる。本言語モデル変換装置においては、ｃｌａｓｓと呼ばれる要素のみを変換するものとする。ｃｌａｓｓは、ｂｅｈａｖｉｏｒとｃｈａｎｎｅｌの２種類ある。本言語モデル変換装置では、ｃｌａｓｓはいずれもＶＨＤＬのＥｎｔｉｔｙとＡｒｃｈｉｔｅｃｔｕｒｅの組に対応させる。

【００４１】

（処理２）ＳｐｅｃＣ言語のｃｌａｓｓは、ｐｏｒｔ変数とローカル変数、そして、関数からなる。本言語モデル変換装置では、ｐｏｒｔ変数は対応するＡｒｃｈｉｔｅｃｔｕｒｅのｐｏｒｔ信号に、ローカル変数はＡｒｃｈｉｔｅｃｔｕｒｅのローカル信号に、関数はＡｒｃｈｉｔｅｃｔｕｒｅのｐｒｏｃｅｓｓにそれぞれ対応させる。

【００４２】

（処理３）ＳｐｅｃＣ言語のｃｌａｓｓにおけるｐｏｒｔ変数には、入力属性（ｉｎ）、出力属性（ｏｕｔ）、入出力属性（ｉｎｏｕｔ）の３種類の方向属性がある。上記の処理２において、ｐｏｒｔ変数のｐｏｒｔ信号への置き換えを、方向属性によって、次のように行う。

・入力属性（ｉｎ）の場合、ＩＮ属性の信号（合計１つ）を対応させる。

・出力属性（ｏｕｔ）の場合、ＯＵＴ属性の信号（合計１つ）を対応させる。

・入出力属性（ｉｎｏｕｔ）の場合、ＩＮ属性の信号とＯＵＴ属性の信号（合計２つ）を対応させる。

【００４３】

（処理４）ＳｐｅｃＣ言語のｃｌａｓｓ内の関数に対し、呼び出し用の信号として、実行トリガー信号（ｐｒｏｃｅｓｓにとっての入力信号）、実行状態信号（ｐｒｏｃｅｓｓにとっての出力信号）を定義する。ｃｌａｓｓの外から呼ばれる関数に関してはＥｎｔｉｔｙのｐｏｒｔ信号として定義する。関数に引数や戻

り値があれば、それに対応する入出力信号を同様にｐｏｒｔに定義する。

【００４４】

（処理５）ｐｒｏｃｅｓｓの冒頭において、外部からの関数の呼び出しに対応するコードを次のように生成する。

```
process
begin
  実行状態信号<= ‘0’ ;
  wait until 実行ﾄﾘｶﾞｰ信号= ‘1’ ;
  実行状態信号<= ‘1’ ;
  wait until CLK= ‘1’  and CLK’ EVENT;
  ---関数内の処理
end process
```

（処理６）関数内のコードについて、関数内のローカル変数はＶＨＤＬのｐｒｏｃｅｓｓ変数に対応させる。変数の代入文は代入文に、ｗａｉｔはｗａｉｔ文に対応させる。ｉｆ等の制御構文はＶＨＤＬのｉｆ等に対応させる。関数の呼び出しは、上記の処理４で生成した信号を用いて次のようなコードを対応させる。

```
実行トリガー信号<= ‘1’ ;
wait until 実行状態信号= ‘1’ ;
実行トリガー信号<= ‘0’ ;
wait until 実行状態信号= ‘0’ ;
```

＜コード置換処理（コード置換部２）＞

（処理７）ＶＨＤＬのｐｒｏｃｅｓｓブロック内部における信号代入は、その場で即座に信号値に反映されるわけではない。反映されるのは、ｗａｉｔ文が処理されるときである。その効果を考慮して、次のように変換する。ＶＨＤＬの信号に対して、一時ｐｒｏｃｅｓｓ変数を対応して定義する。ｗａｉｔ文の直後に、変数：＝信号；の代入文を作り、ｗａｉｔ文の直前に、信号＜＝変数；の代入文を作る。ＳｐｅｃＣ言語の関数外変数への代入文は一時変数への代入文とする

。また、値の参照も一時変数の値の参照とする。

【００４５】

一例として、ａを関数外変数として、次のようなＳｐｅｃＣ言語コードを考える。

```
waitfor 100;
a=1;
a=a+1;
waitfor 100;
```

これに対応するＶＨＤＬコードは次のようになる。ａがＳｐｅｃＣ変数ａに対応する信号であり、ｔｍｐ＿ａがその信号に対応する一時ｐｒｏｃｅｓｓ変数である。

```
wait for 100 ns;
tmp_a:=a;
tmp_a:=1;
tmp_a:=tmp_a+1;
a<=tmp_a;
wait for 100 ns;
```

ただし、代入先がＳｐｅｃＣ言語のｐｏｒｔ変数に対応する場合、代わりにＯＵＴ属性の信号に代入する。ローカル信号の場合は、特に変えなくて良い。

【００４６】

ＳｐｅｃＣ言語の変数の型がｅｖｅｎｔであった場合、特に一時変数は作らない。

【００４７】

イベントが起こったことを表す方法は、イベント信号が１クロック間‘1’の値をとることとする。イベント変数に関してはｎｏｔｉｆｙ　ｅ；が代入に相当する。

```
notify e;
waitfor 100;
```

は、次のようなコードに対応させる。

```
e<= '1' ;
wait until CLK= '1'  and CLK' EVENT ;
c<= '0' ;
wait for 100 ns;
```

イベントの参照は、ｗａｉｔ　ｅ；で行われるが、このコードに対応するＶＨＤＬコードは、

```
wait e= '1' ;
```

となる。

【００４８】

（処理８）ＳｐｅｃＣ言語のｃｌａｓｓは他のクラスをインスタンス化することができる。インスタンス化はＶＨＤＬのｐｏｒｔ＿ｍａｐ宣言に対応させる。ただし、ｐｏｒｔが上記の処理３で複数に分裂している場合があるので、ローカル変数に接続している場合は、同じ信号を、ポート変数を接続している場合は、同じ方向属性の信号を接続する。

【００４９】

例えば、次のようなＳｐｅｃＣ言語コードを考える。

```
behavior B0(inout int b);
behavior B1(inout int a){
  B2 aB2(a);
  // …
};
```

これに対応するＶＨＤＬコードは次のようになる。

```
Entity B1 is port(a:in integer; a_out:  ut integer; … );end B1;
Architecture B1 od B1 is
Component B0 is port(b: in integer; b_out: out integer; … );end c
omponent;
begin
  aB2: B2 port map(a,a_out,…);
  ――  …
end B1;
```

＜値解決プロセス生成処理（値解決プロセス生成部３）＞

（処理９）各Ａｒｃｈｉｔｅｃｔｕｒｅ内で、複数のｐｒｏｃｅｓｓやｐｏｒｔ＿ｍａｐからの代入が起こっている信号を検出する。

検出した信号ｓｉｇに対して、（ａ）代入しようとするプロセスｆに対してデータ信号ｓｉｇ＿ｏｕｔ＿ｆと代入タイミング信号ｓｉｇ＿ｏｔｒｉｇ＿ｆをＡｒｃｈｉｔｅｃｔｕｒｅローカル信号として定義する。

（ｂ）（ａ）で生成した信号を入力とし、信号ｓｉｇを出力とする値解決プロセスを生成する。

プロセスの実装するアルゴリズムは、図８のようである。

（ｃ）各プロセスのｓｉｇへの代入文ｓｉｇ＜＝式；を次のように置き換える。

```
sig_out_f<=式;
sig_otrig<= ‘1’ ;
wait until CLK= ‘1’  and CLK’ EVENT;
sig_otrig<= ‘0’ ;
```

関数実行制御用の信号の値解決プロセスに関しては、実行トリガーと引数に関する処理をのをひとまとめで行う。実行トリガー信号は代入タイミング信号と同様な役目を負わせることができるからである。

【００５０】

（処理１０）上記の処理９で生成した値解決プロセスの出力先の信号がＥｎｔｉｔｙ　Ｐｏｒｔの場合、処理９で、代入タイミング信号を生成している場合がある。その場合は、値解決プロセスの出力として、代入タイミング信号を追加し、値解決プロセスが処理して、出力先にデータを出力するタイミングで、代入タイミング信号の値を変化させるような処理を追加する。

【００５１】

以上の手順によって、例えば図４のような構造のＳｐｅｃＣ言語コードから図５のような構造のＶＨＤＬコードが生成される。

【００５２】

次に、本言語モデル変換装置の他の処理例について説明する。

【００５３】

図９に、本言語モデル変換装置の処理手順の他の例を示す。

【００５４】

ＳｐｅｃＣ言語コードは一般にｃｌａｓｓの間に木構造が形成されている。従って、ＳｐｅｃＣ言語コードを入力して、図６の手順を行って出力したＶＨＤＬコードのＥｎｔｉｔｙ－Ａｒｃｈｉｔｅｃｔｕｒｅ組間にも同様な木構造が形成される。すなわち、図６の手順の値解決プロセス生成処理（処理９、処理１０）で生成した値解決プロセスについて、一般にその木構造に沿って値解決プロセスの木構造ができる。

【００５５】

図９の手順では、まず図６と同様の手順を行い、その後に、値解決プロセス生成部３で後述する値解決プロセス除去処理（処理１１，１２）を行って、プログラムが階層化されている場合に値解決プロセスを１つにするものである。例えば、図６の手順で作られた値解決プロセス（図５参照）のうち最も上位にある値解決プロセスに、階層化されている値解決プロセスを統合する。この結果、例えば、図１０のようになる。値解決プロセス除去処理では、階層の最も上に位置するものだけを残して後は除去するために、下位層に含まれる代入データ信号と代入タイミング信号を各コンポーネントのポート信号にして上位のコンポーネントに接続する。

【００５６】

以下では、値解決プロセス除去処理について説明する。

【００５７】

＜値解決プロセス除去処理（値解決プロセス生成部３）＞

（処理１１）出力に代入タイミング信号がない値解決プロセスが、値解決プロセスの階層の頂上にある。それ以外を「除去可能な値解決プロセス」と呼ぶものとする。除去可能な値解決プロセスに関して、そのプロセスを除去する。

【００５８】

（処理１２）上記の処理１１で除去されたプロセスの入力をＥｎｔｉｔｙポートに展開する。除去されたプロセスの出力ポートは除去する。これを値解決プロセスの下の階層から順に行う。その中間階層の値解決プロセスを処理している場合、ｐｏｒｔ＿ｍａｐの入力信号ポート部が別のポート群に置き換わっているがその置き換わった信号群を入力信号としてみなして処理を行う。最終的に頂上の値解決プロセスの入力は、全ての階層のその信号への代入処理を持つプロセスからのデータ信号と代入タイミング信号になる。

【００５９】

以上説明してきたように、本実施形態によれば、並列プログラムとその間に共有する変数を含むソフトウェア記述を同様の動作をする動作レベルのハードウェア記述に変換することができる。

【００６０】

また、並列プロセスが階層化されている場合にも対応できる。なお、値解決プロセスを階層構造にする構成と、最上位の値解決プロセスのみ残す構成とのいずれかを選択できるようにしてもよい（前者の場合は、処理１０まで行い、後者の場合は、処理１２まで行えばよい）。選択の基準としては、例えば、ポート信号の数を減らしたいときは前者の構成を選択し、階層が深く余分な待ち処理を入れたくない場合は後者の構成を選択するようにする。

【００６１】

図１１に、本発明の一実施形態に係る言語モデル変換装置と呼ぶの他の構成例を示す。この言語モデル変換装置は、図１の構成に値解決プロセス変換部４を付

加したものである。

【００６２】

まず、図１２（ハードウェア記述言語モデルの説明図）を参照して、値解決プロセス変換処理の概要について説明する。なお、図１２において、２０１５は呼び出す側のプロセス、２０１６は呼び出される側のプロセス、２０１０は（呼び出される側の）プロセスへの実行起動トリガー信号、２０１１は（呼び出す側の）プロセスからの実行状態信号、２０１２はプロセスを呼び出すための手続き呼出切り替えプロセス（値解決プロセス変換処理により値解決プロセスが変換されたもの）、２０１３は（呼び出す側の）プロセスから他のプロセスを呼び出すための実行起動トリガー信号、２０１４は（呼び出す側の）プロセスから他のプロセスを呼び出すときの実行状態信号である。

【００６３】

ソフトウェア記述言語モデルにおける並列プログラム間の手続き呼び出しを、ハードウェア記述言語モデルにおいて次のように表現する。

【００６４】

並列プログラムから並列プロセスに変換される手続きを呼び出すコードに対して、次のようなコードを生成する。

・手続きを呼び出す際には、呼び出す側のプロセス２０１５が出力するトリガ信号２０１３（２０１１）と、呼び出される側のプロセス２０１６が出力する実行状態信号２０１１（２０１４）を使う。加えて、呼び出す側のプロセスが出力する引数信号と、呼び出される側のプロセスが出力する返り値信号を使うこともある。

（１）トリガは実際に呼び出される側のプロセスが実行されるまで立ち上げておくとする。

（２）呼び出される側のプロセスに対して、（出力）トリガ信号、引数、（入力）実行状態、返り値の信号の切替を行う手続き呼び出し切替プロセスを作る。

（３）呼び出される側のプロセス対する並列プロセスごとに（呼び出しプロセスに対して出力）トリガ信号、引数、（呼び出しプロセスに対して入力）実行状

態、返り値の信号を別に用意する。

【００６５】

手続き呼び出し切替プロセス２０１２は、次のような各入力トリガの立ち上がり条件で実行する処理を逐次に行う（図１５参照）。

各入力トリガの立ち上がり条件で実行する処理：

（１）他の実行状態信号を下げる。

（２）出力トリガを立てる。同時に、（あれば）処理プロセスからの引数信号値を出力引数信号の値とする。

（３）入力実行状態信号が立ち上がったら、処理プロセスの実行状態信号を立ち上がらせる。

（４）入力実行状態信号が立ち下がると、処理プロセスの状態信号を立ち下がらせる。同時に、（あれば）入力返り値信号値を処理プロセスの返り値信号値にする。

【００６６】

また、ここでは、図４のように入力されるソフトウェア記述言語モデルにおいて手続きが階層化されていて出力されるハードウェア記述言語モデルにおいて並列プロセスが階層化されている場合、図１４のように各階層ごとに１のプロセスに手続き呼び出し切替プロセスを定義する。ただし、切替える必要のない場合は、手続き呼び出し切替プロセスを介さず結線する。

【００６７】

図１３に、本言語モデル変換装置の処理手順を示す。図１３の手順では、まず図６と同様の手順を行う。ソフトウェア記述言語モデルにおける並列プログラム間の手続き呼び出しも、並列プログラム間の共有変数へのアクセスとして扱われるので、図６と同様の手順によって、並列プログラム間の手続き呼び出しについて、値解決プロセスが生成される。次いで、図１３の手順では、値解決プロセス変換部４で後述する値解決プロセス変換処理（処理１３）を行って、値解決プロセス（図３参照）のうち並列プログラム間の手続き呼び出しに該当するものを、手続き呼び出し切り替えプロセス（図１２参照）に変換する（なお、並列プログラム間の手続き呼び出しに該当しないものは、値解決プロセスとして残る）。

【００６８】

以下では、値解決プロセス変換処理について説明する。

【００６９】

＜値解決プロセス変換処理（値解決プロセス変換部４）＞

（処理１３）骨格コード生成処理の処理４で生成したｐｒｏｃｅｓｓの制御用信号（実行トリガー信号、実行状態信号、加えて、引数、戻り値データ信号）の制御信号伝達のための値解決プロセスを、値解決プロセス生成処理の処理９で作成したものから取り替える。具体的には次のような置き換えを行う。

【００７０】

（処理１３－１）

まず、信号の定義の追加を行う。実行状態信号は、各呼び出し側のｐｒｏｃｅｓｓに対して共通であったものを、呼び出しプロセスごとに用意する。同様に、返り値の信号も呼び出しｐｒｏｃｅｓｓ毎に追加定義する。

【００７１】

（処理１３－２）

次に、呼び出し側プロセスにおける信号の置き換えを行う。すなわち、ｗａｉｔ　ｕｎｔｉｌの条件内の実行状態信号を、処理１３－１で生成した信号に置き換える。戻り値信号参照部分に関しても同様に置き換える。

【００７２】

（処理１３－３）

次に、実行トリガー信号の値解決プロセスを、次のようなものに置き換える。

【００７３】

ｐｒｏｃｅｓｓの入力は、各呼び出しｐｒｏｃｅｓｓから実行トリガー信号と引数のデータ信号と、加えて、呼び出されるｐｒｏｃｅｓｓから出力される実行状態信号及び返り値信号、出力は、各呼び出しｐｒｏｃｅｓｓ信号用の実行状態信号と返り値信号と、加えて、呼び出されるｐｒｏｃｅｓｓへの実行トリガー信号及び引数データ信号である。

【００７４】

さらに、このｐｒｏｃｅｓｓは、図１５に示すような動作またはこれと等価な

動作を実装する。例えば、実装する動作は、次の通りである。

（１）実行トリガーを監視する無限ループがある。このループを抜ける条件は、入力の実行トリガー信号の少なくとも１つの値が立ち上がることである。

（２）ループを抜けた直後に、各実行トリガー信号が立ち上がっているという条件の条件分岐を用意する。ただし、この条件分岐は、次のように逐次直列に並んでいるものとする。

```
if 条件1 then
  処理1;
end if;
if 条件2 then
  処理2;
end if;
…
```

（３）各条件が成り立ったときの処理は、次のようである。

（３－１）呼び出されるｐｒｏｃｅｓｓへの実行トリガー信号値を立ち上げる。

（３－２）呼び出されるｐｒｏｃｅｓｓからの実行状態信号が立ち上がるのを待つ。

（３－３）呼び出しｐｒｏｃｅｓｓへの実行状態信号を立ち上げる。

（３－４）呼び出されるｐｒｏｃｅｓｓからの実行状態信号が立ち下がるのを待つ。

（３－５）呼び出しｐｒｏｃｅｓｓへのの実行状態信号が立ち下げ、呼び出されるｐｒｏｃｅｓｓからの返り値信号値を呼び出しｐｒｏｃｅｓｓの戻り値信号に伝達する。

【００７５】

次に、本言語モデル変換装置の他の処理例について説明する。

【００７６】

図１６に、本言語モデル変換装置の処理手順の他の例を示す。

【００７７】

この手順は、図１３の骨格コード生成処理（処理１～６）、コード置換生成処理（処理７，８）、値解決プロセス生成処理（処理９，１０）を行い、そして、値解決プロセス除去処理（処理１１，１２）を行った後に、図１３の値解決プロセス変換処理（処理１３）を行うものである。

【００７８】

すなわち、並列プロセスが階層化されている場合、階層的に生成される手続き呼び出し切替プロセスのうち最も上位に位置するものを残して他の手続き切替プロセスは生成しないようにする。下位のプロセスへの入力出力信号を残したプロセスの入力出力信号とする。そのために必要なポート信号を生成する。

【００７９】

この結果、例えば、図１４は、図１７のようになる。

【００８０】

以上説明してきたように、本実施形態によれば、ソフトウェア言語による並列プログラムをハードウェア記述に変換する際、手続き呼び出しに相当する排他的なプロセス実行処理を実現した（排他的呼び出しを可能にした）ハードウェア動作記述に変換することができる。

【００８１】

また、並列プロセスが階層化されている場合にも対応できる。なお、手続き呼び出し解決プロセスを階層構造にする構成と、最上位の手続き呼び出し解決プロセスのみ残す構成とのいずれかを選択できるようにしてもよい（前者の場合は、処理１０までと処理１３を行い、後者の場合は、処理１３まで行えばよい）。選択の基準としては、例えば、ポート信号の数を減らしたいときは前者の構成を選択し、階層が深く余分な遅延を入れたくない場合は後者の構成を選択するようにする。

【００８２】

なお、以上の各機能は、ソフトウェアとして実現可能である。

また、本実施形態は、コンピュータに所定の手段を実行させるための（あるいはコンピュータを所定の手段として機能させるための、あるいはコンピュータに

所定の機能を実現させるための）プログラムとして実施することもでき、該プログラムを記録したコンピュータ読取り可能な記録媒体として実施することもできる。

【００８３】

なお、この発明の実施の形態で例示した構成は一例であって、それ以外の構成を排除する趣旨のものではなく、例示した構成の一部を他のもので置き換えたり、例示した構成の一部を省いたり、例示した構成に別の機能あるいは要素を付加したり、それらを組み合わせたりすることなどによって得られる別の構成も可能である。また、例示した構成と論理的に等価な別の構成、例示した構成と論理的に等価な部分を含む別の構成、例示した構成の要部と論理的に等価な別の構成なども可能である。また、例示した構成と同一もしくは類似の目的を達成する別の構成、例示した構成と同一もしくは類似の効果を奏する別の構成なども可能である。

また、この発明の実施の形態で例示した各種構成部分についての各種バリエーションは、適宜組み合わせて実施することが可能である。

また、この発明の実施の形態は、個別装置としての発明、関連を持つ２以上の装置についての発明、システム全体としての発明、個別装置内部の構成部分についての発明、またはそれらに対応する方法の発明等、種々の観点、段階、概念またはカテゴリに係る発明を包含・内在するものである。

従って、この発明の実施の形態に開示した内容からは、例示した構成に限定されることなく発明を抽出することができるものである。

【００８４】

本発明は、上述した実施の形態に限定されるものではなく、その技術的範囲において種々変形して実施することができる。

【００８５】

【発明の効果】

本発明によれば、並列プログラムとその間に共有する変数を含むソフトウェア記述を同様の動作をする動作レベルのハードウェア記述に変換するができる。

【００８６】

また、本発明によれば、ソフトウェア言語による並列プログラムをハードウェア記述に変換する際、手続き呼び出しに相当する排他的なプロセス実行処理を実現することができる。

【図面の簡単な説明】

【図１】

本発明の実施の形態に係るソフトウェア・ハードウェア言語モデル変換装置の構成例を示す図

【図２】

変数を共有する並列ソフトウェアプログラムのソフトウェア記述の一例について説明するための図

【図３】

変数を共有する並列ソフトウェアプログラムのソフトウェア記述を変換して得られる値解決プロセスを含むハードウェア記述の一例について説明するための図

【図４】

変数を共有する階層化された並列ソフトウェアプログラムのソフトウェア記述の一例について説明するための図

【図５】

変数を共有する階層化された並列ソフトウェアプログラムのソフトウェア記述を変換して得られる階層化された値解決プロセスを含むハードウェア記述の一例について説明するための図

【図６】

同実施形態に係るソフトウェア・ハードウェア言語モデル変換装置の処理手順の一例を示すフローチャート

【図７】

同実施形態に係るソフトウェア・ハードウェア言語モデル変換装置の処理手順の一例を示すフローチャート

【図８】

値解決プロセスによる処理の一例を示すフローチャート

【図９】

同実施形態に係るソフトウェア・ハードウェア言語モデル変換装置の処理手順の他の例を示すフローチャート

【図１０】

変数を共有する階層化された並列ソフトウェアプログラムのソフトウェア記述を変換して得られる最上位階層以外の値解決プロセスを削除したハードウェア記述の一例について説明するための図

【図１１】

同実施形態に係るソフトウェア・ハードウェア言語モデル変換装置の他の構成例を示す図

【図１２】

値解決プロセスを含むハードウェア記述を変換して得られる手続き呼び出し切り替えプロセスを含むハードウェア記述の一例について説明するための図

【図１３】

同実施形態に係るソフトウェア・ハードウェア言語モデル変換装置の処理手順のさらに他の例を示すフローチャート

【図１４】

階層化された値解決プロセスを含むハードウェア記述を変換して得られる階層化された手続き呼び出し切り替えプロセスを含むハードウェア記述の一例について説明するための図

【図１５】

手続き呼び出し切り替えプロセスによる処理の一例を示すフローチャート

【図１６】

同実施形態に係るソフトウェア・ハードウェア言語モデル変換装置の処理手順のさらに他の例を示すフローチャート

【図１７】

最上位階層以外の値解決プロセスを削除したハードウェア記述を変換して得られる最上位階層以外の手続き呼び出し切り替えプロセスを削除したハードウェア記述の一例について説明するための図

【図１８】

同実施形態に係るシステム設計支援装置の構成例を示す

【符号の説明】

１…骨格コード生成部

２…コード置換部

３…値解決プロセス生成部

４…値解決プロセス変換部

１００…システム設計支援装置

１０１…仕様モデル記述部

１０２…アーキテクチャ探索部

１０３…コミュニケーション合成部

１０４…ハードウェア仕様生成部

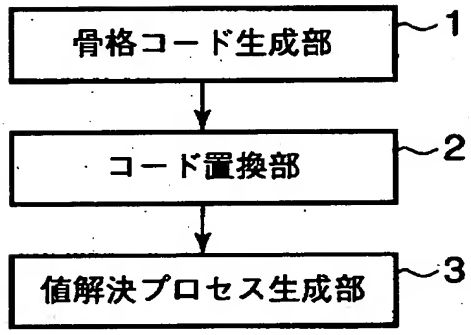
１０５…部品化・再利用部

１０６…ソフトウェア仕様生成部

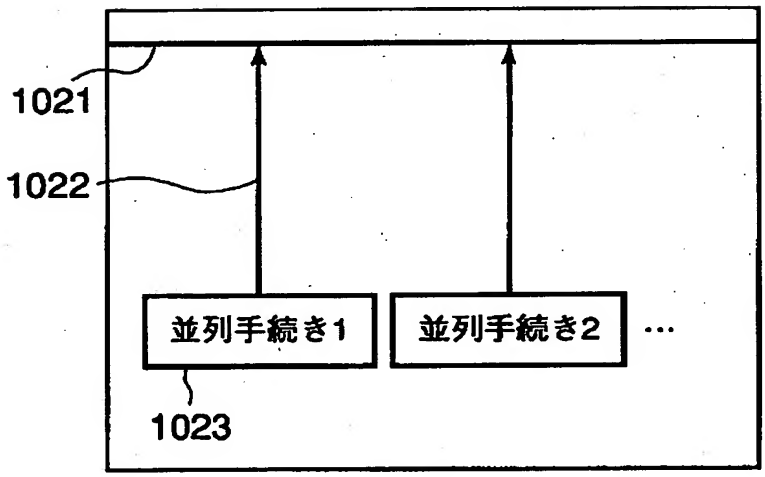
１０７…システム仕様記録部

【書類名】　図面

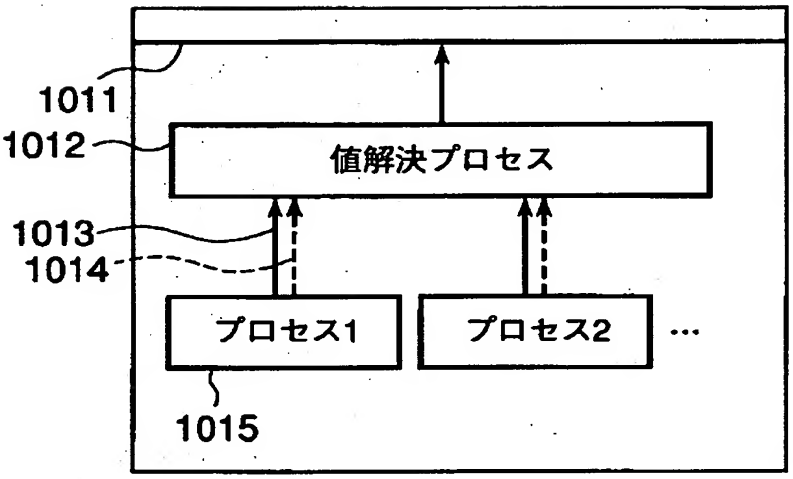
【図１】


骨格コード生成部
1
コード置換部
2
値解決プロセス生成部
3

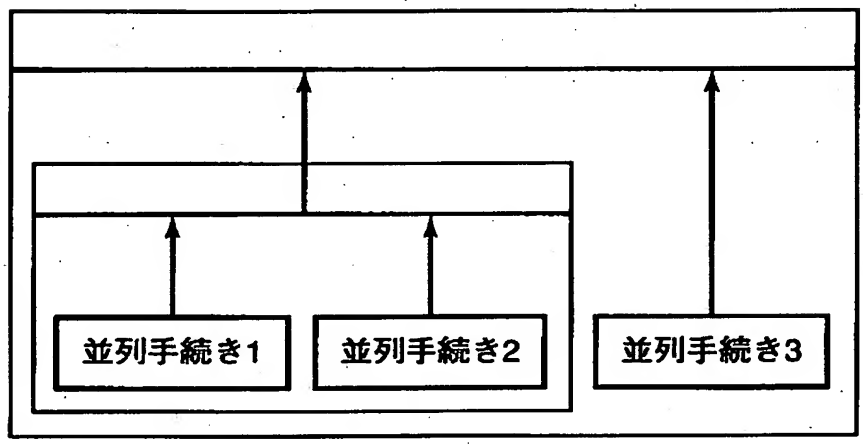
【図２】


1021
1022
並列手続き1
並列手続き2
…
1023

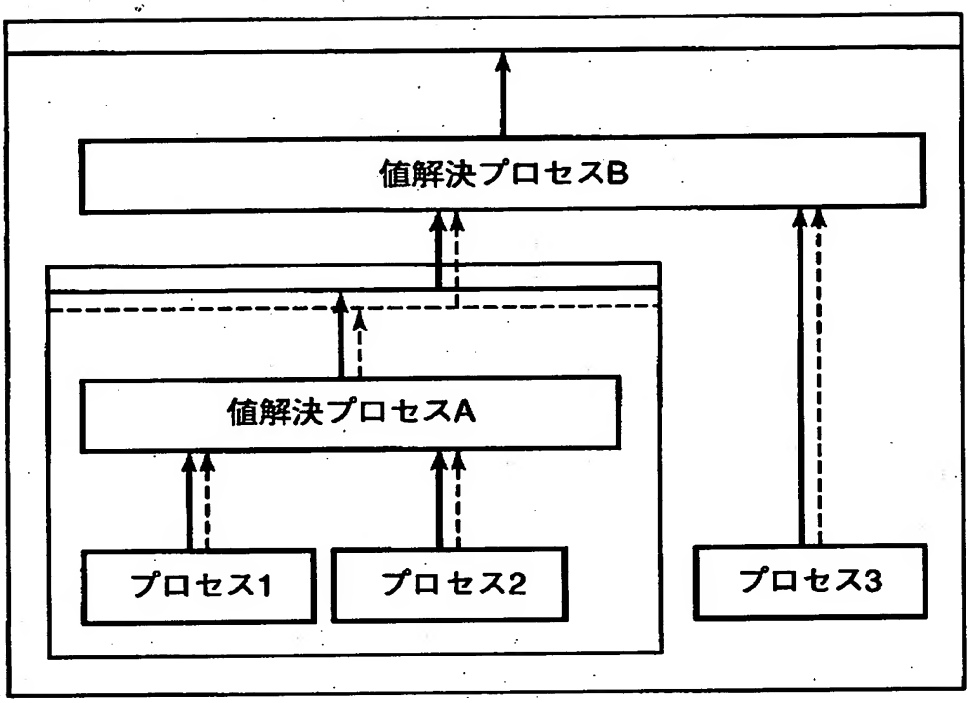
【図３】


1011
1012
値解決プロセス
1013
1014
プロセス1
プロセス2
…
1015

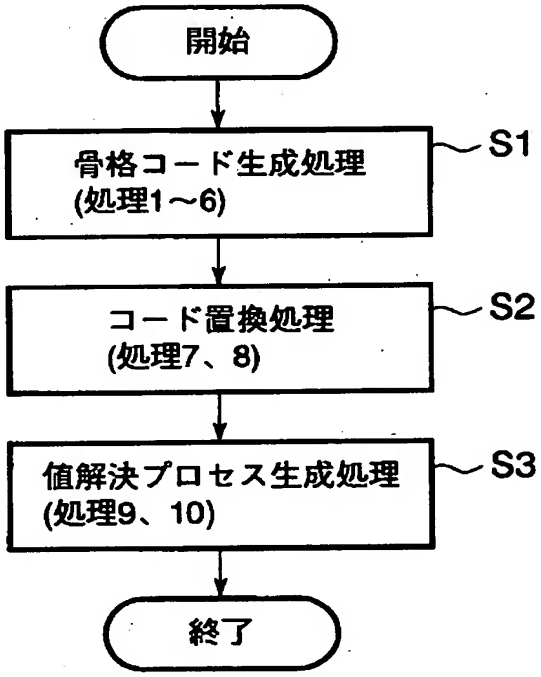
【図４】


並列手続き1
並列手続き2
並列手続き3

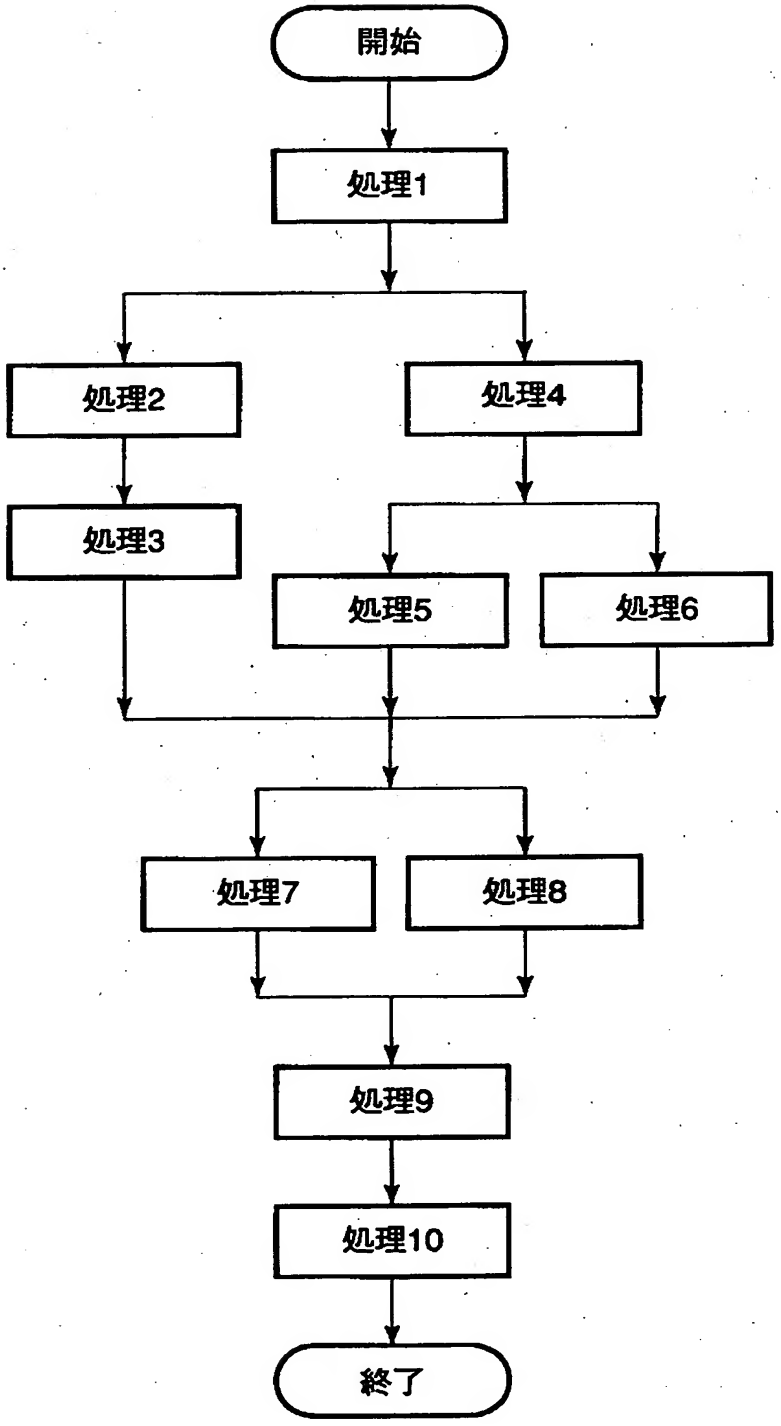
【図５】


値解決プロセスB
値解決プロセスA
プロセス1
プロセス2
プロセス3

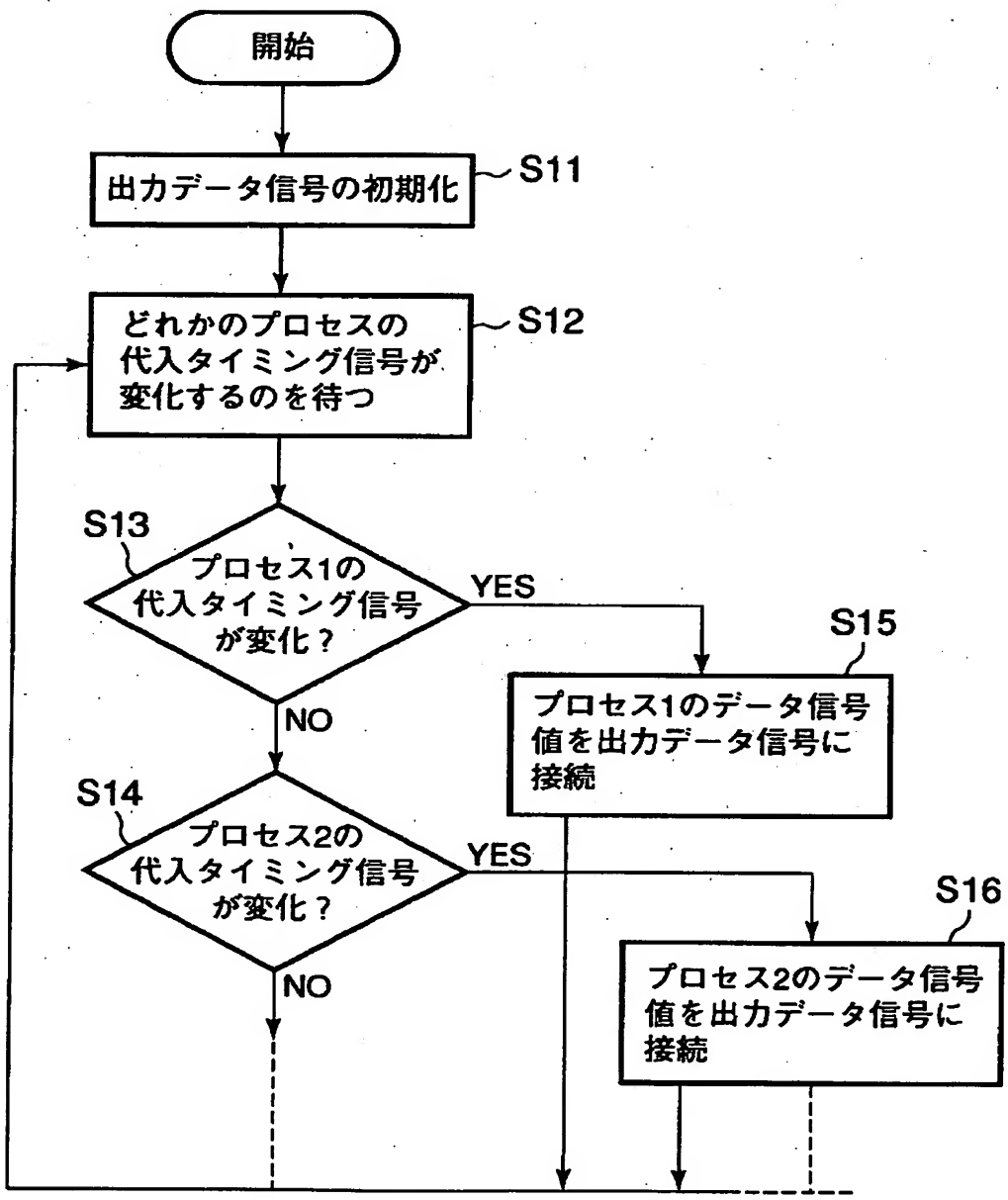
【図６】


開始
骨格コード生成処理
(処理1～6)
S1
コード置換処理
(処理7、8)
S2
値解決プロセス生成処理
(処理9、10)
S3
終了

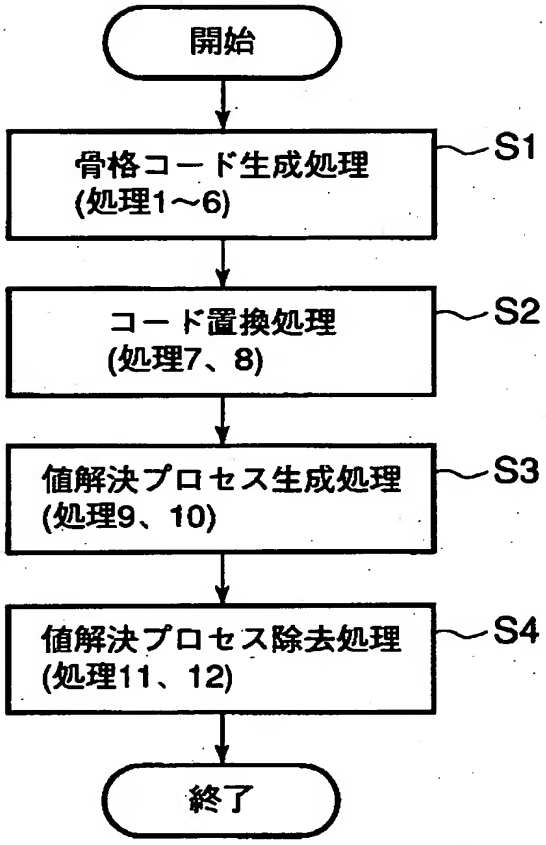
【図７】



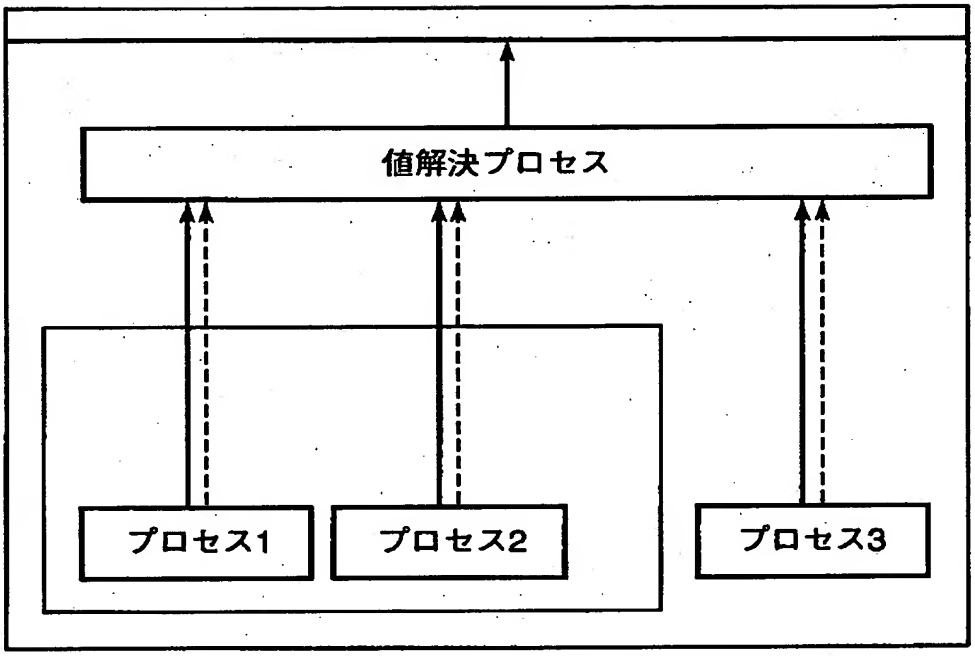
【図８】


開始
出力データ信号の初期化
S11
どれかのプロセスの代入タイミング信号が変化するのを待つ
S12
S13
プロセス1の代入タイミング信号が変化？
YES
NO
S15
プロセス1のデータ信号値を出力データ信号に接続
S14
プロセス2の代入タイミング信号が変化？
YES
NO
S16
プロセス2のデータ信号値を出力データ信号に接続

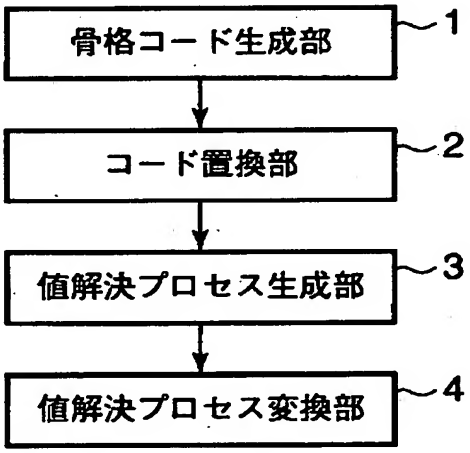
【図９】


開始
骨格コード生成処理
(処理1～6)
S1
コード置換処理
(処理7、8)
S2
値解決プロセス生成処理
(処理9、10)
S3
値解決プロセス除去処理
(処理11、12)
S4
終了

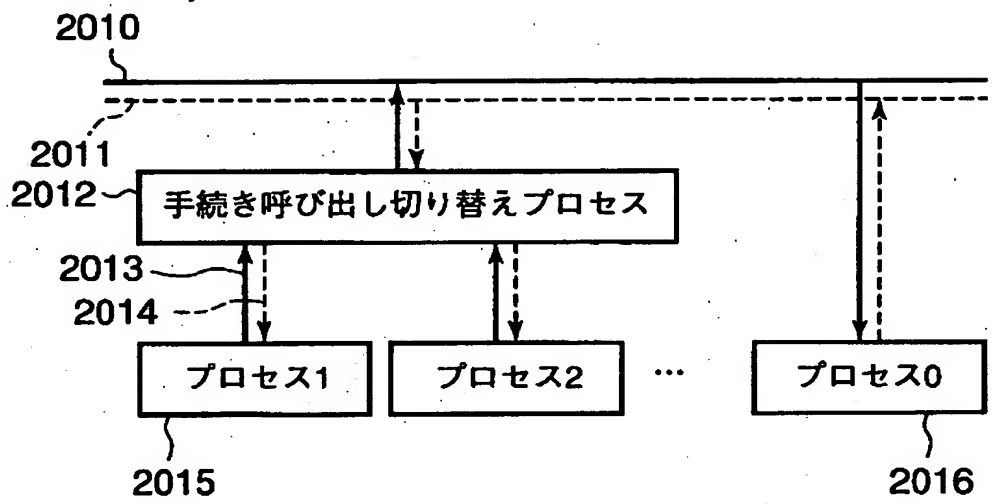
【図１０】


値解決プロセス
プロセス1
プロセス2
プロセス3

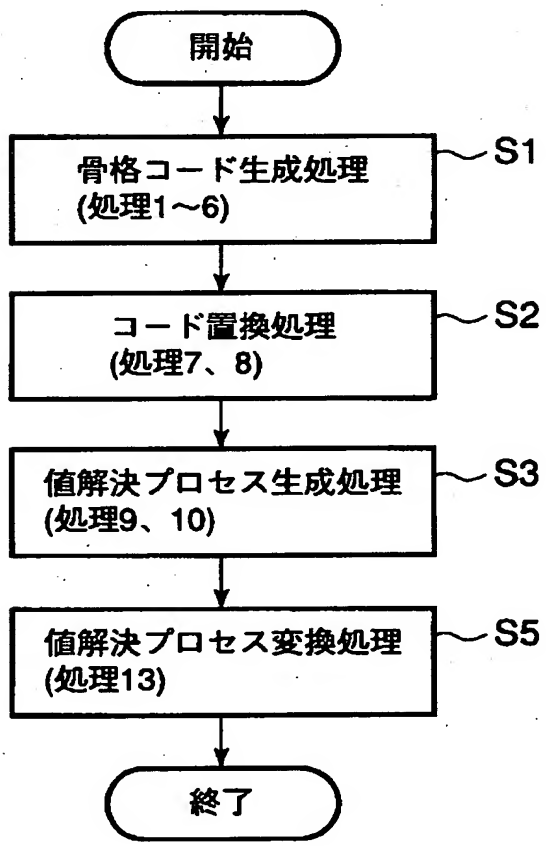
【図１１】


骨格コード生成部
1
コード置換部
2
値解決プロセス生成部
3
値解決プロセス変換部
4

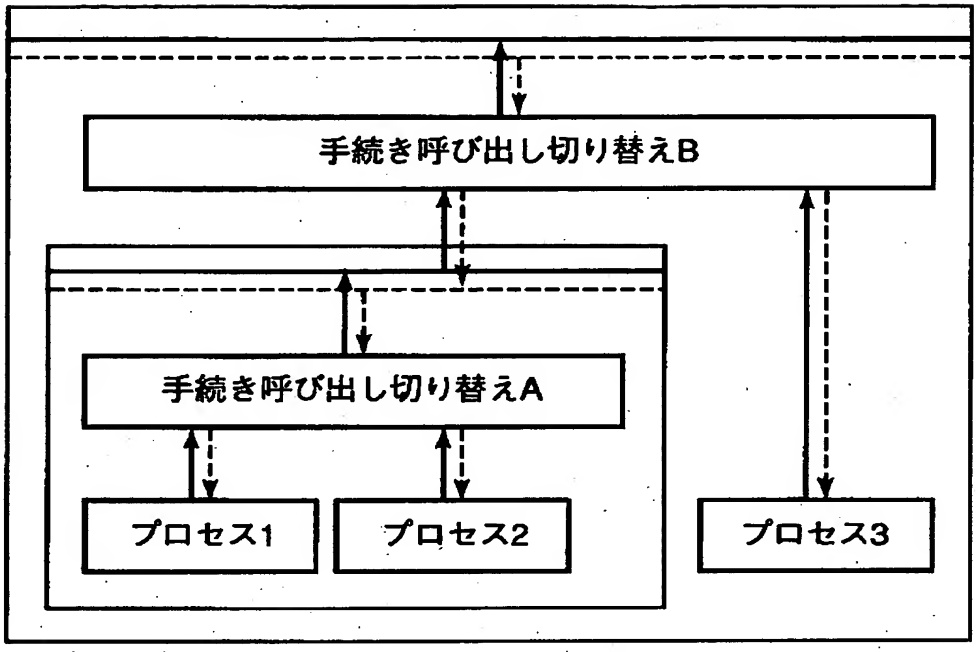
【図１２】


2010
2011
2012
手続き呼び出し切り替えプロセス
2013
2014
プロセス1
プロセス2
…
プロセス0
2015
2016

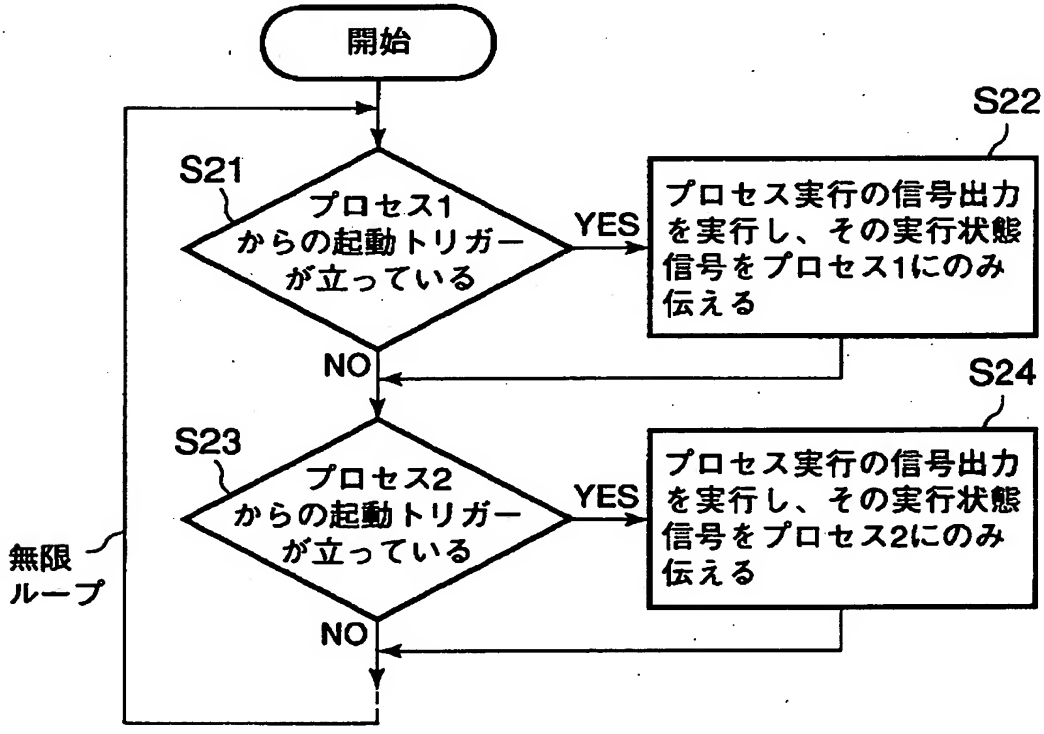
【図１３】


開始
骨格コード生成処理
(処理1～6)
S1
コード置換処理
(処理7、8)
S2
値解決プロセス生成処理
(処理9、10)
S3
値解決プロセス変換処理
(処理13)
S5
終了

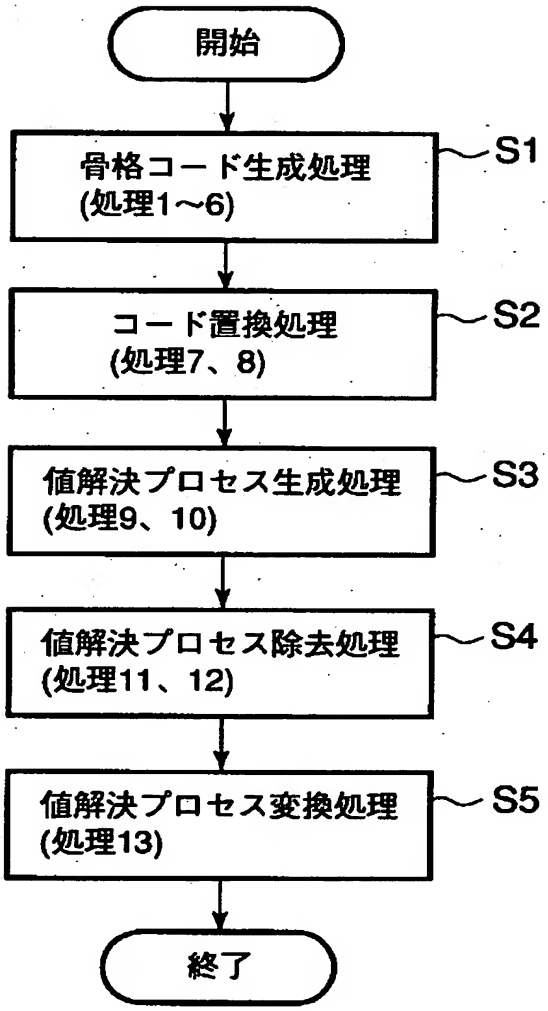
【図１４】


手続き呼び出し切り替えB
手続き呼び出し切り替えA
プロセス1
プロセス2
プロセス3

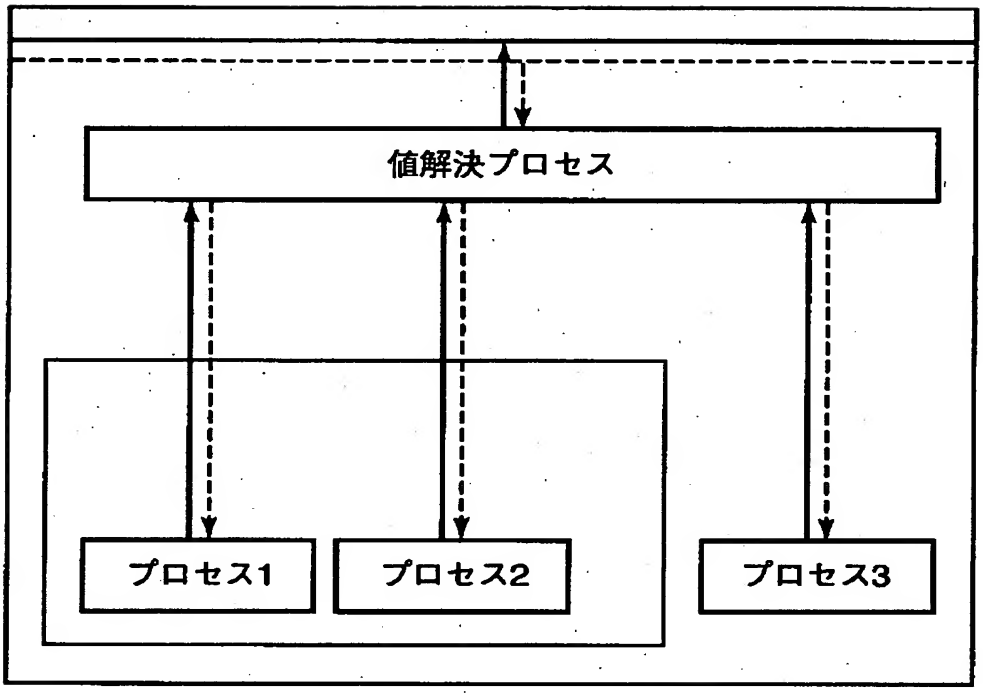
【図１５】


開始
S21
プロセス1からの起動トリガーが立っている
YES
S22
プロセス実行の信号出力を実行し、その実行状態信号をプロセス1にのみ伝える
NO
S23
プロセス2からの起動トリガーが立っている
YES
S24
プロセス実行の信号出力を実行し、その実行状態信号をプロセス2にのみ伝える
NO
無限ループ

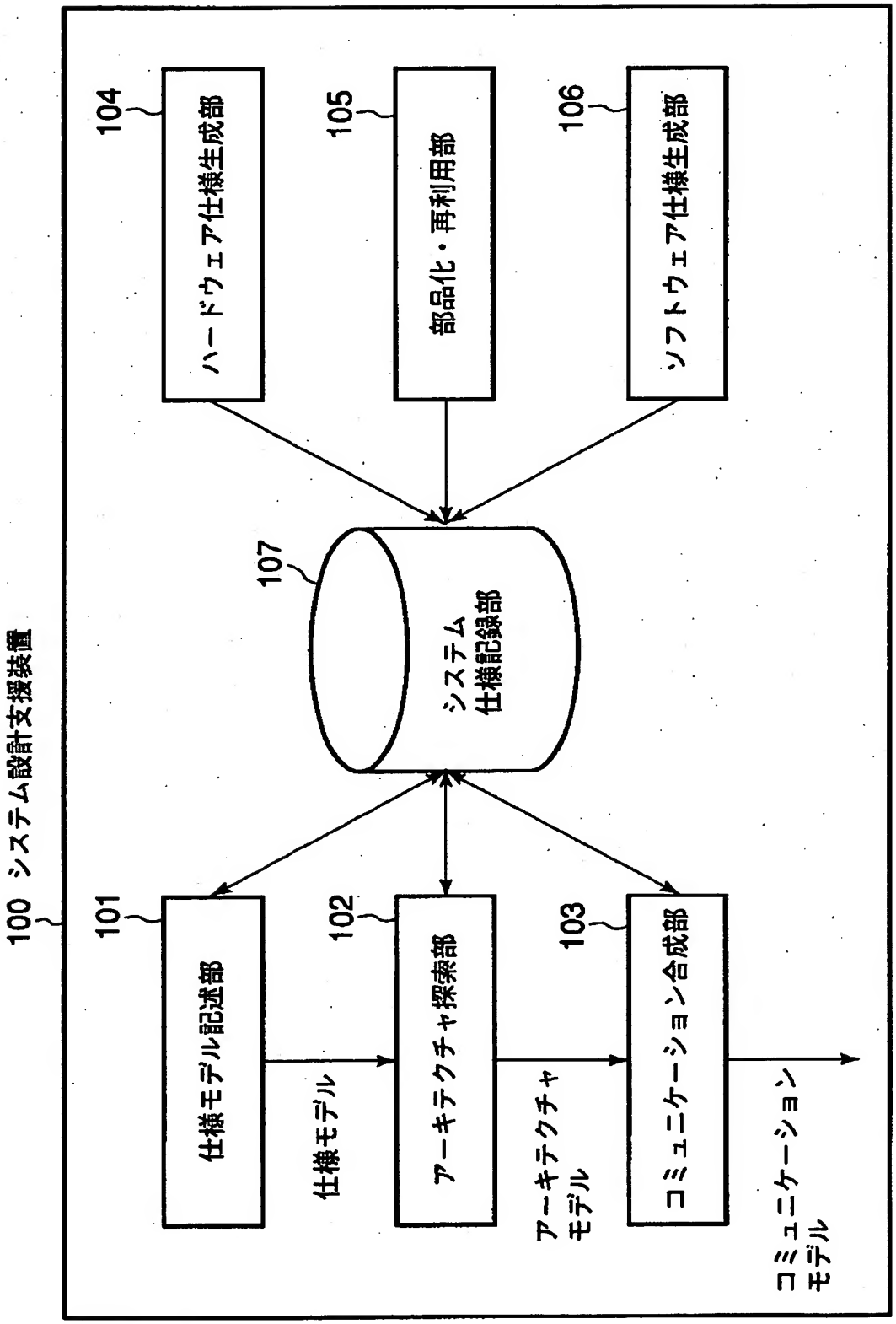
【図１６】


開始
骨格コード生成処理
(処理1～6)
S1
コード置換処理
(処理7、8)
S2
値解決プロセス生成処理
(処理9、10)
S3
値解決プロセス除去処理
(処理11、12)
S4
値解決プロセス変換処理
(処理13)
S5
終了

【図１７】



【図１８】


100 システム設計支援装置
101
仕様モデル記述部
102
アーキテクチャ探索部
103
コミュニケーション合成部
104
ハードウェア仕様生成部
105
部品化・再利用部
106
ソフトウェア仕様生成部
107
システム
仕様記録部
仕様モデル
アーキテクチャ
モデル
コミュニケーション
モデル

【書類名】　　要約書

【要約】

【課題】　並列プログラムとその間に共有する変数を含むソフトウェア記述を同様の動作をする動作レベルのハードウェア記述言語記述に変換することのできるソフトウェア・ハードウェア言語モデル変換装置を提供すること。

【解決手段】　ソフトウェア記述言語による第１のモデルに同一の共有変数へ書き込みを行う複数の並列手続きが含まれているか否かを考慮せずに該第１のモデルをハードウェア記述言語による第２のモデルに変換する。これにより得られた第２のモデルに、同一の共有変数へ書き込みを行う複数の並列手続きに相当する複数の並列プロセスが存在するか否か検出する。検出された並列プロセスの全部又は一部の各プロセスから、データ信号及び代入タイミング信号の組を入力し、該データ信号のうち該代入タイミング信号が変化したプロセスに対応するものを、共有変数の値を保持する信号へ出力する値解決プロセスを生成する。

【選択図】　　図３

# 出願人履歴情報

識別番号　〔０００００３０７８〕

１．変更年月日　１９９０年　８月２２日
［変更理由］　新規登録
住　所　神奈川県川崎市幸区堀川町７２番地
氏　名　株式会社東芝

２．変更年月日　２００１年　７月　２日
［変更理由］　住所変更
住　所　東京都港区芝浦一丁目１番１号
氏　名　株式会社東芝